

Table of Contents

SQLITE DATABASE OBJECT MANAGEMENT	2
SQLITE TABLES	3
<i>SQLite Table Designer</i>	6
SQLite Table Fields	7
Setting SQLite Table Field Properties	9
Setting Other SQLite Table Field Properties	11
SQLite Table Indexes	13
Setting SQLite Table Index Properties	14
SQLite Table Foreign Keys	15
Setting SQLite Table Foreign Key Properties	16
SQLite Table Uniques	18
Setting SQLite Table Unique Properties	19
SQLite Table Checks	22
Setting SQLite Table Check Properties	23
SQLite Table Triggers	24
Setting SQLite Table Trigger Properties	25
SQLite Table Options	26
SQLITE VIEWS	28
<i>SQLite View Designer</i>	31
Working with SQLite View Builder (Available only in Full Version)	32
Editing SQLite View SQL Definition	33
SQLite View Preview	34
SQLite View Explain	35
<i>SQLite View Viewer</i>	36
SQLITE INDEXES	37
<i>SQLite Index Designer</i>	39
Editing SQLite Index General	40
SQLITE TRIGGERS	41
<i>SQLite Trigger Designer</i>	43
Editing SQLite Trigger General	44
Editing SQLite Trigger Definition	45


SQLite Database Object Management

The following list contains the most common SQLite database objects supported by Navicat.

- [Tables](#)
- [Views](#)
- [Indexes](#)
- [Triggers](#)



SQLite Tables


Relational databases use tables to store data. All operations on data are done on the tables themselves or produce another tables as the result. A table is a set of rows and columns, and their intersections are fields. From a general perspective, columns within a table describe the name and type of data that will be found by row for that column's fields. Rows within a table represent records composed of fields that are described from left to right by their corresponding column's name and type. Each field in a row is implicitly correlated with each other field in that row.

Just simply click  to open an object pane for **Table**. A right-click displays the popup menu or using the object pane toolbar, allowing you to create new, edit, open and delete the selected table.

Create Table

To create a new table

- Select anywhere on the object pane.
- Click the  **New Table** from the object pane toolbar.
or
- Right-click and select  **New Table** from the popup menu.
- Edit table properties and fields on the appropriate tabs of the Table Designer.




Hint: To create new table you can also right-click the Tables node of the navigation pane and select the  **New Table** from the popup menu.

To create a new table with the same properties as one of the existing tables has (using popup menu)

Apply to: current database {same connection}



- Select the table(s) for copying in the navigation pane/object pane.
- Right-click and select the **Duplicate Table** from the popup menu.
- The newly created table(s) will be named as "tablename_**copy**".

To create a new table with modification as one of the existing tables

- Select the table for modifying in the navigation pane/object pane.
- Right-click and select the  **Design Table** from the popup menu.
or
- Click the  **Design Table** from the object pane toolbar.
- Modify table properties and fields on the appropriate tabs of the Table Designer.
- Click  **Save As**.

Edit Table

To edit the existing table (manage its fields, indexes, foreign keys and triggers etc)



- Select the table for editing in the navigation pane/object pane.
- Right-click and select the  **Design Table** from the popup menu.
or
- Click the  **Design Table** from the object pane toolbar.
- Edit table properties and fields on the appropriate tabs of the Table Designer.


To change the name of the table

- Select the table for editing in the navigation pane/object pane.
- Right-click and select the **Rename** from the popup menu.


Open Table (manage table data)

To open a table

- Select the table for opening in the navigation pane/object pane.
- Right-click and select the  **Open Table** from the popup menu or simply double-click the table.
or
- Click the  **Open Table** from the object pane toolbar.

Note: This option is only applied if you do wish Navicat loads all your images while opening the table. To open the graphical table with faster performance, use  **Open Table (Quick)** below.

To open a table with graphical fields

- Select the table for opening in the navigation pane/object pane.
- Right-click and select the  **Open Table (Quick)** from the popup menu.

Note: Faster performance for opening the graphical table, as BLOB fields (images) will not be loaded until you click on the cell.



Empty Table

To empty a table

- Select the table in the navigation pane/object pane.
- Right-click the selected table and choose **Empty Table** from the popup menu.

Delete Table

To delete a table

- Select the table for deleting in the navigation pane/object pane.
- Right-click and select the  **Delete Table** from the popup menu.
or
- Click the  **Delete Table** from the object pane toolbar.
- Confirm deleting in the dialog window.

Achieve Table Information

To achieve a table information

- Select the table in the navigation pane/object pane.
- Right-click the selected table and choose **Object Information** from the popup menu.
or
- Choose View -> Object Information in the main menu.

SQLite Table Designer

Table Designer is the basic Navicat tool for working with tables. It allows you to create, edit and drop table's fields, indexes, foreign keys, and much more.



- [Managing Table Fields](#)
- [Managing Table Indexes](#)
- [Managing Table Foreign Keys](#)
- [Managing Table Uniques](#)
- [Managing Table Checks](#)
- [Managing Table Triggers](#)
- [Managing Table Options](#)
- Table SQL Preview

SQLite Table Fields

Table fields are managed on the **Fields** tab of the Table Designer. Just simply click a field for editing. A right-click displays the popup menu or using field toolbar, allowing you to create new, insert, move and drop the selected field.

Add Field

To add a field to the table



- Open the table in the Table Designer.
- Open the **Fields** tab.
- Right-click and select the  **Add Field** from the popup menu or click the  **Add Field** from the toolbar.
- Edit field properties.

To add a new field with modification as one of the existing fields

- Open the table in the Table Designer.
- Open the **Fields** tab.
- Select field.
- Right-click and select the **Duplicate Field** from the popup menu.
- Edit field properties.

Insert Field

To insert a field above an existing field





- Open the table in the Table Designer.
- Open the **Fields** tab.
- Select field.
- Right-click and select the  **Insert Field** from the popup menu or click the  **Insert Field** from the toolbar.
- Define field properties in the empty row.

Edit Field

To edit the table field



- Open the table in the Table Designer.
- Open the **Fields** tab.
- Simply click on the field to edit.

To change the order of the table fields


- Open the table in the Table Designer.
- Open the **Fields** tab.
- Right-click on the field to move and select the  **Move Up**/  **Move Down** from the popup menu or click the  **Move Up**/  **Move Down** from the toolbar.

Delete Field

To delete the table field

- Open the table in the Table Designer.
- Open the **Fields** tab.
- Right-click on the field to delete and select the  **Delete Field** from the popup menu or click the  **Delete Field** from the toolbar.
- Confirm deleting in the dialog window.

Setting SQLite Table Field Properties

Name	Type	Length	Decimals	Allow Null	
▶ OrderNo	INTEGER	0	0	<input type="checkbox"/>	 1
CustNo	INTEGER	0	0	<input checked="" type="checkbox"/>	
SaleDate	TEXT	0	0	<input checked="" type="checkbox"/>	
ShipDate	TEXT	0	0	<input checked="" type="checkbox"/>	
EmpNo	INTEGER	0	0	<input checked="" type="checkbox"/>	
ShipToContact	TEXT	0	0	<input checked="" type="checkbox"/>	

Name

The Name is a descriptive identifier for a field that can be up to 64 characters (letters or numbers) including spaces. The names should be descriptive enough that anyone can easily identify them when viewing or editing records. For example, LastName, FirstName, StreetAddress, or HomePhone.

Use the **Name** edit box to set the field name. Note that the name of the field must be unique among all the field names in the table.

Type

The **Type** dropdown list defines the type (storage class) of the field data.

The following tables summarize each type:

Type	Description
INTEGER	The value is a signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value.
REAL	The value is a floating point value, stored as an 8-byte IEEE floating point number.
TEXT	The value is a text string, stored using the database encoding (UTF-8, UTF-16BE or UTF-16LE).
BLOB	The value is a blob of data, stored exactly as it was input.

Note for SQLite Version 2:

1. You can store any kind of data you want in any column of any table, regardless of the declared datatype of that column.

[Click here](#) for detailed description on datatype in SQLite version 2.

Note for SQLite Version 3:

1. *Storage class is slightly more general than a datatype. The INTEGER storage class, for example, includes 6 different integer datatypes of different lengths.*
2. *In order to maximize compatibility between SQLite and other database engines, SQLite supports the concept of "type affinity" on columns.*

[Click here](#) for detailed description on datatype, storage class and type affinity.

Length and Decimals

Use the **Length** edit box to define the length of the field and use **Decimals** edit box to define the number of digits after the decimal point (the scale).

Allow Null

Allow the NULL values for the field.

Primary Key

A **Primary Key** is a single field or combination of fields that uniquely defines a record. None of the fields that are part of the primary key can contain a null value.

Setting Other SQLite Table Field Properties

Default

To set the default value for the field.

Collation

To specify the text collating function to use when comparing text entries for the column. The built-in BINARY collating function is used by default.

BINARY

Compares string data using `memcmp()`, regardless of text encoding.

NOCASE

The same as binary, except the 26 upper case characters of ASCII are folded to their lower case equivalents before the comparison is performed. Note that only ASCII characters are case folded. SQLite does not attempt to do full UTF case folding due to the size of the tables required.

RTRIM

The same as binary, except that trailing space characters are ignored.

Note: Support in SQLite 3.

Not null ON CONFLICT

To specify an algorithm used to resolve constraint conflicts if Allow Null option is unchecked. The default conflict resolution algorithm is ABORT.

ROLLBACK

When a constraint violation occurs, an immediate ROLLBACK occurs, thus ending the current transaction, and the command aborts with a return code of `SQLITE_CONSTRAINT`. If no transaction is active (other than the implied transaction that is created on every command) then this algorithm works the same as ABORT.

ABORT

When a constraint violation occurs, the command backs out any prior changes it might have made and aborts with a return code of `SQLITE_CONSTRAINT`. But no ROLLBACK is executed so changes from prior commands within the same transaction are preserved. This is the default behavior.

FAIL

When a constraint violation occurs, the command aborts with a return code `SQLITE_CONSTRAINT`. But any changes to the database that the command made prior to encountering the constraint violation are preserved and are not backed out. For example, if an `UPDATE` statement encountered a constraint violation on the 100th row that it attempts to update, then the first 99 row changes are preserved but changes to rows 100 and beyond never occur.

IGNORE

When a constraint violation occurs, the one row that contains the constraint violation is not inserted or changed. But the command continues executing normally. Other rows before and after the row that contained the constraint violation continue to be inserted or updated normally. No error is returned when the `IGNORE` conflict resolution algorithm is used.

REPLACE

When a `UNIQUE` constraint violation occurs, the pre-existing rows that are causing the constraint violation are removed prior to inserting or updating the current row. Thus the insert or update always occurs. The command continues executing normally following `REPLACE`. No error is returned by the `REPLACE` conflict resolution. If a `NOT NULL` constraint violation occurs, the `NULL` value is replaced by the default value for that column. If the column has no default value, then the `ABORT` algorithm is used. If a `CHECK` constraint violation occurs then the `IGNORE` algorithm is used.

Auto Increment (INTEGER only)

The `AUTO INCREMENT` attribute can be used to generate a unique identity for new rows. To start with the `AUTO INCREMENT` value other than 1, you can set that value in Options tab.



SQLite Table Indexes

Index provides a faster access path to table data. It is created using one or more columns of a table to speed SQL statement execution on that table.

Table indexes are managed on the **Indexes** tab of the Table Designer. Just simply click/double-click an index field for editing. A right-click displays the popup menu or using the index toolbar, allowing you to create new, edit and delete the selected index field.

Add Index

To add a table index

- Open the table in the Table Designer.
- Open the **Indexes** tab.
- Right-click and select the  **Add Index** from the popup menu or click the  **Add Index** from the toolbar.
- Edit index properties.



Edit Index

To edit a table index

- Open the table in the Table Designer.
- Open the **Indexes** tab.
- Just simply click/double-click on the index to edit.

Delete Index

To delete a table index

- Open the table in the Table Designer.
- Open the **Indexes** tab.
- Right-click on the index to delete and select the  **Delete Index** from the popup menu or click the  **Delete Index** from the toolbar.
- Confirm deleting in the dialog window.

Setting SQLite Table Index Properties

Name	Fields	Unique
▶ OrderIndex	OrderNo, CustNo	... <input type="checkbox"/>

Use the **Name** edit box to set the index name.

To include field(s) in the index, just simply double-click the **Fields**

Select the field(s) from the list. To remove the fields from the index, uncheck them in the same way. You can also use the arrow buttons to change the index field(s) order.

Collation

To define a collating sequence used for text entries in that column. The default collating sequence is the collating sequence defined for that column.

BINARY

Compares string data using memcmp(), regardless of text encoding.

NOCASE

The same as binary, except the 26 upper case characters of ASCII are folded to their lower case equivalents before the comparison is performed. Note that only ASCII characters are case folded. SQLite does not attempt to do full UTF case folding due to the size of the tables required.

RTRIM

The same as binary, except that trailing space characters are ignored.

Note: Support in SQLite 3.

Sort Order

To indicate sort order - ascending "ASC" or descending "DESC".

Unique

All values of the indexed column(s) must only occur once.



SQLite Table Foreign Keys

A foreign key is a field in a relational table that matches the primary key column of another table.

Foreign Keys are managed on the **Foreign Keys** tab of the Table Designer. Just simply click/double-click a foreign key field for editing. A right-click displays the popup menu or using the foreign key toolbar, allowing you to create new, edit and delete the selected foreign key field.

Add Foreign Key

To add a foreign key

- Open the table in the Table Designer.
- Open the **Foreign Keys** tab.
- Right-click and select the  **Add Foreign Key** from the popup menu or click the  **Add Foreign Key** from the toolbar.
- Edit foreign key properties.



Edit Foreign Key

To edit a foreign key

- Open the table in the Table Designer.
- Open the **Foreign Keys** tab.
- Just simply click/double-click on the foreign key to edit.

Delete Foreign Key

To delete a foreign key


- Open the table in the Table Designer.
- Open the **Foreign Keys** tab.
- Right-click on the foreign key to delete and select the  **Delete Foreign Key** from the popup menu or click the  **Delete Foreign Key** from the toolbar.
- Confirm deleting in the dialog window.

Setting SQLite Table Foreign Key Properties

Name	Fields	Reference Table	Reference Fields	On Delete	On Update
▶ cust_order_fk	CustNo	⋮ customer	CustNo	NO ACTION	NO ACTION

Use the **Name** edit box to enter a name for the new key and then select a table field to include in the key from the **Fields** group.

Use the **Reference Table** dropdown list to select a foreign table respectively.

To include field(s) to the key, just simply double-click the **Fields/Reference Fields** field or click  to open the editor(s) for editing.

The **On Delete** and **On Update** dropdown list define the type of the actions to be taken.

RESTRICT

The "RESTRICT" action means that the application is prohibited from deleting (for ON DELETE RESTRICT) or modifying (for ON UPDATE RESTRICT) a parent key when there exists one or more child keys mapped to it.

NO ACTION

Configuring "NO ACTION" means just that: when a parent key is modified or deleted from the database, no special action is taken.

CASCADE

A "CASCADE" action propagates the delete or update operation on the parent key to each dependent child key. For an "ON DELETE CASCADE" action, this means that each row in the child table that was associated with the deleted parent row is also deleted. For an "ON UPDATE CASCADE" action, it means that the values stored in each dependent child key are modified to match the new parent key values.

SET NULL

If the configured action is "SET NULL", then when a parent key is deleted (for ON DELETE SET NULL) or modified (for ON UPDATE SET NULL), the child key columns of all rows in the child table that mapped to the parent key are set to contain SQL NULL values.

SET DEFAULT

The "SET DEFAULT" actions are similar to "SET NULL", except that each of the child key columns is set to contain the columns default value instead of NULL.

Deferred

Deferred foreign key constraints are not checked until the transaction tries to COMMIT.



SQLite Table Uniques

Unique constraints ensure that the data contained in a column or a group of columns is unique with respect to all the rows in the table.

Uniques are managed on the **Uniques** tab of the Table Designer. Just simply click/double-click an unique field for editing. Using the unique toolbar, allowing you to create new, edit and delete the selected unique field.

Add Unique

To add an unique

- Open the table in the Table Designer.
- Open the **Uniques** tab.
- Right-click and select the  **Add Unique** from the popup menu or click the  **Add Unique** from the toolbar.
- Edit unique properties.



Edit Unique

To edit an unique

- Open the table in the Table Designer.
- Open the **Uniques** tab.
- Just simply click on the unique to edit.


Delete Unique

To delete an unique


- Open the table in the Table Designer.
- Open the **Uniques** tab.
- Right-click on the unique to delete and select the  **Delete Unique** from the popup menu or click the  **Delete Unique** from the toolbar.
- Confirm deleting in the dialog window.

Setting SQLite Table Unique Properties

Use the **Name** edit box to set the unique name.

Name	Fields
▶ OrderUnique	OrderNo, CustNo 

Fields

To set field(s) as unique, just simply double-click the **Fields** field or click  to open the editor(s) for editing.

Collation

To define a collating sequence used for text entries in that column. The default collating sequence is the collating sequence defined for that column.

BINARY

Compares string data using memcmp(), regardless of text encoding.

NOCASE

The same as binary, except the 26 upper case characters of ASCII are folded to their lower case equivalents before the comparison is performed. Note that only ASCII characters are case folded. SQLite does not attempt to do full UTF case folding due to the size of the tables required.

RTRIM

The same as binary, except that trailing space characters are ignored.

Note: Support in SQLite 3.

Sort Order

To indicate sort order - ascending "ASC" or descending "DESC".

ON CONFLICT

To specify an algorithm used to resolve constraint conflicts. The default conflict resolution algorithm is ABORT.

ROLLBACK

When a constraint violation occurs, an immediate ROLLBACK occurs, thus ending the current transaction, and the command aborts with a return code of

SQLITE_CONSTRAINT. If no transaction is active (other than the implied transaction that is created on every command) then this algorithm works the same as ABORT.

ABORT

When a constraint violation occurs, the command backs out any prior changes it might have made and aborts with a return code of SQLITE_CONSTRAINT. But no ROLLBACK is executed so changes from prior commands within the same transaction are preserved. This is the default behavior.

FAIL

When a constraint violation occurs, the command aborts with a return code SQLITE_CONSTRAINT. But any changes to the database that the command made prior to encountering the constraint violation are preserved and are not backed out. For example, if an UPDATE statement encountered a constraint violation on the 100th row that it attempts to update, then the first 99 row changes are preserved but changes to rows 100 and beyond never occur.

IGNORE

When a constraint violation occurs, the one row that contains the constraint violation is not inserted or changed. But the command continues executing normally. Other rows before and after the row that contained the constraint violation continue to be inserted or updated normally. No error is returned when the IGNORE conflict resolution algorithm is used.

REPLACE

When a UNIQUE constraint violation occurs, the pre-existing rows that are causing the constraint violation are removed prior to inserting or updating the current row. Thus the insert or update always occurs. The command continues executing normally following REPLACE. No error is returned by the REPLACE conflict resolution. If a NOT NULL constraint violation occurs, the NULL value is replaced by the default value for that column. If the column has no default value, then the ABORT algorithm is used. If a CHECK constraint violation occurs then the IGNORE algorithm is used.

SQLite Table Checks



A check constraint allows you to specify that the value in a certain column must satisfy a Boolean (truth-value) expression.

Note: Checks are supported from SQLite version 3.3.0 or later.

Checks are managed on the **Checks** tab of the Table Designer. Just simply click/double-click a check field for editing. Using the check toolbar, allowing you to create new, edit and delete the selected check field.

Add Check

To add a check

- Open the table in the Table Designer.
- Open the **Checks** tab.
- Right-click and select the  **Add Check** from the popup menu or click the  **Add Check** from the toolbar.
- Edit check properties.



Edit Check

To edit a check

- Open the table in the Table Designer.
- Open the **Checks** tab.
- Just simply click on the check to edit.

Delete Check

To delete a check

- Open the table in the Table Designer.
- Open the **Checks** tab.
- Right-click on the check to delete and select the  **Delete Check** from the popup menu or click the  **Delete Check** from the toolbar.
- Confirm deleting in the dialog window.

Setting SQLite Table Check Properties

Use the **Name** edit box to set the check name.

Check

Set the condition for checking, e.g. "field_name1 > 0 AND field_name2 > field_name1" in the **Check** edit box.

Definition

Type in the definition for the check constraint.



SQLite Table Triggers

A trigger is a database operation that is automatically performed when a specified database event occurs.

Triggers are managed on the **Triggers** tab of the Table Designer. Just simply click a trigger field for editing. A right-click displays the popup menu or using the trigger toolbar, allowing you to create new, edit and delete the selected trigger field.

Add Trigger

To add a trigger

- Open the table in the Table Designer.
- Open the **Triggers** tab.
- Right-click and select the  **Add Trigger** from the popup menu or click the  **Add Trigger** from the toolbar.
- Edit trigger properties.



Edit Trigger

To edit a trigger

- Open the table in the Table Designer.
- Open the **Triggers** tab.
- Just simply click on the trigger to edit.

Delete Trigger

To delete a trigger

- Open the table in the Table Designer.
- Open the **Triggers** tab.
- Right-click on the trigger to delete and select the  **Delete Trigger** from the popup menu or click the  **Delete Trigger** from the toolbar.
- Confirm deleting in the dialog window.

Setting SQLite Table Trigger Properties

Name

Set the trigger name.

Fires

Determine when the trigger actions will be executed relative to the insertion, modification or removal of the associated row.

INSERT

Fires the trigger whenever an INSERT statement adds a row to a table.

UPDATE

Fires the trigger whenever an UPDATE statement changes a value in one of the columns specified in **Update of Fields**. If no **Update of Fields** are present, the trigger will be fired whenever an UPDATE statement changes a value in any column of the table.

DELETE

Fires the trigger whenever a DELETE statement removes a row from the table.

Update of Fields

Specify the fields for UPDATE statement trigger upon necessary.

Definition

Type in the definition for the trigger.

Advanced

When Clause

Specify the trigger condition, which is a SQL condition that must be satisfied for the database to fire the trigger.

SQLite Table Options

Table Options are managed on the **Options** tab of the Table Designer. Just simply click an option for editing.

Primary Key ON CONFLICT

To specify an algorithm used to resolve primary key constraint conflicts. The default conflict resolution algorithm is ABORT.

ROLLBACK

When a constraint violation occurs, an immediate ROLLBACK occurs, thus ending the current transaction, and the command aborts with a return code of SQLITE_CONSTRAINT. If no transaction is active (other than the implied transaction that is created on every command) then this algorithm works the same as ABORT.

ABORT

When a constraint violation occurs, the command backs out any prior changes it might have made and aborts with a return code of SQLITE_CONSTRAINT. But no ROLLBACK is executed so changes from prior commands within the same transaction are preserved. This is the default behavior.

FAIL

When a constraint violation occurs, the command aborts with a return code SQLITE_CONSTRAINT. But any changes to the database that the command made prior to encountering the constraint violation are preserved and are not backed out. For example, if an UPDATE statement encountered a constraint violation on the 100th row that it attempts to update, then the first 99 row changes are preserved but changes to rows 100 and beyond never occur.

IGNORE

When a constraint violation occurs, the one row that contains the constraint violation is not inserted or changed. But the command continues executing normally. Other rows before and after the row that contained the constraint violation continue to be inserted or updated normally. No error is returned when the IGNORE conflict resolution algorithm is used.

REPLACE


When a UNIQUE constraint violation occurs, the pre-existing rows that are causing the constraint violation are removed prior to inserting or updating the current row. Thus the insert or update always occurs. The command continues executing normally following REPLACE. No error is returned by the REPLACE conflict resolution. If a NOT NULL constraint violation occurs, the NULL value is replaced by the default value for that column. If the column has no default value, then the ABORT algorithm is used. If a CHECK constraint violation occurs then the IGNORE algorithm is used.

Auto Increment

Set/Reset the **Auto Increment** value in the edit field. The **Auto Increment Value** indicates the value for next record.



SQLite Views


Views are useful for allowing users to access a set of tables as if it were a single table, and limiting their access to just that. Views can also be used to restrict access to rows (a subset of a particular table).

Just simply click  to open an object pane for **View**. A right-click displays the popup menu or using the object pane toolbar, allowing you to create new, edit, open and delete the selected view.




Create View

To create a new view




- Select anywhere on the object pane.
- Click the  **New View** from the object pane toolbar.
or
- Right-click and select  **New View** from the popup menu.
- Edit view properties on the appropriate tabs of the View Designer.

Hint: To create new view you can also right-click the Views node of the navigation pane and select the  **New View** from the popup menu.

To create a new view with modification as one of the existing views



- Select the view for modifying in the navigation pane/object pane.
- Right-click and select the  **Design View** from the popup menu.
or
- Click the  **Design View** from the object pane toolbar.
- Modify view properties on the appropriate tabs of the View Designer.
- Click  **Save As**.

To create a new view with loading from a SQL file

- Select anywhere on the object pane.
- Click the  **New View** from the object pane toolbar.
or
- Right-click and select  **New View** from the popup menu.
- Click  **Load**.

Edit View

To edit the existing view (manage its SQL definition etc)



- Select the view for editing in the navigation pane/object pane.
- Right-click and select the  **Design View** from the popup menu.
or
- Click the  **Design View** from the object pane toolbar.
- Edit view properties on the appropriate tabs of the View Designer.

To change the name of the view

- Select the view for editing in the navigation pane/object pane.
- Right-click and select the **Rename** from the popup menu.



Open View

To open a view (manage view data)

- Select the view for opening in the navigation pane/object pane.
- Right-click and select the  **Open View** from the popup menu or simply double-click the view.
or
- Click the  **Open View** from the object pane toolbar.

Delete View

To delete a view

- Select the view for deleting in the navigation pane/object pane.
- Right-click and select the  **Delete View** from the popup menu.
or
- Click the  **Delete View** from the object pane toolbar.
- Confirm deleting in the dialog window.

Achieve View Information

To achieve a view information

- Select the view in the navigation pane/object pane.
- Right-click the selected view and choose **Object Information** from the popup menu.
or
- Choose View -> Object Information in the main menu.

SQLite View Designer

View Designer is the basic Navicat tool for working with views. It allows you to create new view and edit the existing view definition (view name and the SELECT statement it implements).

- [Working with View Builder](#)
- [Editing View SQL Definition](#)
- View SQL Preview
- [View Preview](#)
- [View Explain](#)

Working with SQLite View Builder (Available only in Full Version)

View Builder allows you to build views visually. It allows you to create and edit views without knowledge of SQL. See Query Builder for details.

Editing SQLite View SQL Definition


The **Definition** tab allows you to edit the view definition as SQL statement (SELECT statement it implements).

Example:

```
SELECT
  clients.RecordID
FROM
  clients
```


Hint: To customize the view of the editor and find out more features for sql editing, see [Editor View and More Features](#).

SQLite View Preview

To preview the result of the view, click  **Preview** on the toolbar. If the query statement is correct, the **Result** and **Message** tabs will be opened.

The **Result** tab displays the data of the view as a grid and the **Message** tab displays the message log.

SQLite View Explain

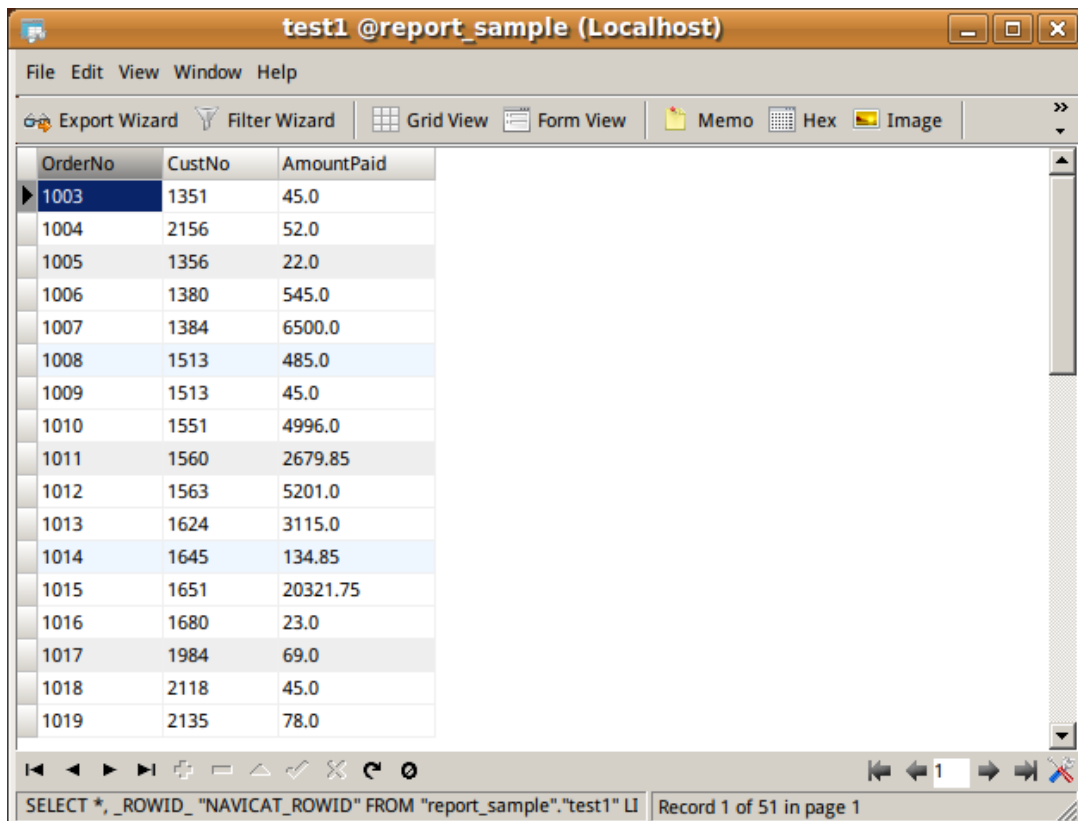
To return information about how the SQL statement would have operated, click  **Explain** on the toolbar. If the query statement is correct, the **Explain** tab will show.

SQLite View Viewer

View Viewer displays the view data as a grid. Data can be displayed in three modes:  **Grid View**,  **Form View** and **Text/Blob View**. See Data View for details.


The toolbars of View Viewer provides the following functions for managing data:

- **Export Data**
Export data to TXT, DBF, HTML, SQL, RTF and more.
- **Filter Data**
Allow you to filter records by creating and applying filter criteria for the data grid.
- **Edit TEXT/BLOB**
Allow you to view and edit the content of TEXT and BLOB fields.





SQLite Indexes

Index provides a faster access path to table data. It is created using one or more columns of a table to speed SQL statement execution on that table.




Just simply click  to open an object pane for **Index**. A right-click displays the popup menu or using the object pane toolbar, allowing you to create new, edit and delete the selected Index.

Create Index

To create a new index



- Select anywhere on the object pane.
- Click the  **New Index** from the object pane toolbar.
or
- Right-click and select  **New Index** from the popup menu.
- Edit index properties on the appropriate tabs of the Index Designer.

To create a new index with modification as one of the existing index

- Select the index for modifying in the object pane.
- Right-click and select the  **Design Index** from the popup menu or simply double-click the index.
or
- Click the  **Design Index** from the object pane toolbar.
- Modify index properties on the appropriate tabs of the Index Designer.
- Click  **Save As**.

Edit Index

To edit the existing index (manage its properties etc)

- Select the index for editing in the object pane.
- Right-click and select the  **Design Index** from the popup menu or simply double-click the index.
or
- Click the  **Design Index** from the object pane toolbar.
- Edit index properties on the appropriate tabs of the Index Designer.

To change the name of the index

- Select the index for editing in the object pane.
- Right-click and select the **Rename** from the popup menu.



Maintain Index

To maintain an index

- Select the index for maintaining in the object pane.
- Right-click and select the **Maintain** from the popup menu.
 - ReIndex

Delete Index

To delete an index

- Select the index for deleting in the object pane.
- Right-click and select the  **Delete Index** from the popup menu.
or
- Click the  **Delete Index** from the object pane toolbar.
- Confirm deleting in the dialog window.

Achieve Index Information

To achieve an index information (Index Owner, Index Type and DDL, etc)

- Select the index in the object pane.
- Right-click the selected index and choose **Index Information** from the popup menu.
or
- Choose View -> Object Information in the main menu.

SQLite Index Designer

Index Designer is the basic Navicat tool for working with indexes. It allows you to create new index and edit the existing index properties.

- [Editing Index General](#)
- Index SQL Preview

Editing SQLite Index General

Type

The types of the index.

Normal

A normal index does not impose restrictions on the column values.

Unique

An unique index indicates that no two rows of a table have duplicate values in the key columns.

Table name

The table that contains the index.

Fields

Name

To define the field.

Collate

To define a collating sequence used for text entries in that column. The default collating sequence is the collating sequence defined for that column.

BINARY

Compares string data using memcmp(), regardless of text encoding.

NOCASE

The same as binary, except the 26 upper case characters of ASCII are folded to their lower case equivalents before the comparison is performed. Note that only ASCII characters are case folded. SQLite does not attempt to do full UTF case folding due to the size of the tables required.

RTRIM

The same as binary, except that trailing space characters are ignored.

Note: Support in SQLite 3.


Sort Order

To indicate sort order - ascending "ASC" or descending "DESC".

SQLite Triggers



Triggers are database operations that are automatically performed when a specified database event occurs.

See [Triggers](#) for details.




Just simply click  to open an object pane for **Trigger**. A right-click displays the popup menu or using the object pane toolbar, allowing you to create new, edit and delete the selected trigger.

Create Trigger

To create a new trigger



- Select anywhere on the object pane.
- Click the  **New Trigger** from the object pane toolbar.
or
- Right-click and select  **New Trigger** from the popup menu.
- Edit trigger properties on the appropriate tabs of the Trigger Designer.

To create a new trigger with modification as one of the existing trigger

- Select the trigger for modifying in the object pane.
- Right-click and select the  **Design Trigger** from the popup menu or simply double-click the trigger.
or
- Click the  **Design Trigger** from the object pane toolbar.
- Modify trigger properties on the appropriate tabs of the Trigger Designer.
- Click  **Save As**.

Edit Trigger

To edit the existing trigger (manage its general, advance, etc)



- Select the trigger for editing in the object pane.
- Right-click and select the  **Design Trigger** from the popup menu or simply double-click the trigger.
or
- Click the  **Design Trigger** from the object pane toolbar.
- Edit trigger properties on the appropriate tabs of the Trigger Designer.

To change the name of the trigger

- Select the trigger for editing in the object pane.
- Right-click and select the **Rename** from the popup menu.

Delete Trigger

To delete a trigger

- Select the trigger for deleting in the object pane.
- Right-click and select the  **Delete Trigger** from the popup menu.
or
- Click the  **Delete Trigger** from the object pane toolbar.
- Confirm deleting in the dialog window.

Achieve Trigger Information

To achieve a trigger information

- Select the trigger in the object pane.
- Right-click the selected trigger and choose **Object Information** from the popup menu.
or
- Choose View -> Object Information in the main menu.

SQLite Trigger Designer

Trigger Designer is the basic Navicat tool for working with triggers. It allows you to create new trigger and edit the existing trigger definition.

- [Editing Trigger General](#)
- [Editing Trigger Definition](#)
- Trigger SQL Preview

Editing SQLite Trigger General

Trigger Type

Define the trigger type: TABLE or VIEW.

Table name or View name

Choose a table or view.

Fires

Determine when the trigger actions will be executed relative to the insertion, modification or removal of the associated row.

When

Specify the trigger condition for the database to fire the trigger.

On Event

It indicates the kind of statement that activates the trigger.

Insert

The trigger is activated whenever adding a row to a table.

Delete

The trigger is activated whenever removing a row from the table.

Update

The trigger is activated whenever changing a value in one of the fields selected in

Update Of Fields.

Update of Fields

Specify the fields for UPDATE statement trigger upon necessary.

Editing SQLite Trigger Definition

The **Definition** tab allows you to edit valid SQL statements in the trigger definition inside *BEGIN* and *END*.